Solution - Exam Preparation Test

1. Pod Logging

Download the YAML file and create a pod with `kubectl apply -f cka_logs.yaml` command

Fetch the logs and store it to a directory with the following command:

```
kubectl logs counter2 --all-containers | grep "02" > /opt/kplabs-foobar
```

2. Daemonset

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kplabs-daemonset
spec:
  selector:
    matchLabels:
      name: kplabs-all-pods
  template:
    metadata:
      labels:
        name: kplabs-all-pods
    spec:
      containers:
      - name: kplabs-pods
        image: nginx
```

3 Init Container:

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: base-pod
  name: base-pod
spec:
  containers:
  - command:
    - sleep
    - "3600"
    image: busybox
    name: base-pod
    volumeMounts:
    - mountPath: /opt
      name: opt-volume
```

```
  restartPolicy: Never
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', 'touch /opt/myfile;']
    volumeMounts:
    - mountPath: /opt
      name: opt-volume
  volumes:
  - name: opt-volume
    emptyDir: {}
```

4. Multi-Container Pods

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  namespace: test
  labels:
    run: nginx
  name: kucc4
spec:
  containers:
  - image: nginx
    name: nginx
  - image: redis
    name: redisd
  - image: memcached
    name: memcached
  - image: consul
    name: consul
  restartPolicy: Never
```

5. NodeSelector

i) Label the node with the one mentioned in the question:

`kubectl label node kubeadm-worker disktype=ssd`

ii) Run the POD on the node with the label which we have previously added.

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: kplabs-selector
  name: kplabs-selector
spec:
  containers:
  - image: nginx
    name: kplabs-selector
```

```
    nodeSelector:
      disktype: ssd
    restartPolicy: Never
```

6. Deployment - Rolling Updates and Rollbacks

i) Create the namespace

```
kubectl create namespace production
```

ii) Create the deployment with three replicas

```
kubectl run nginx-app --image=nginx:1.11.9-alpine --replicas=3
--namespace=production --restart=Always --record
```

iii) Update the image:

```
kubectl -n production set image deployment/nginx-app nginx-app=nginx:1.12.0-alpine
--record
```

iv) Rollback the update:

```
kubectl rollout undo deployment -n production nginx-app --to-revision=1
```

7. Service

```
kubectl -n production expose deployment nginx-app --name=front-end-service
--type=ClusterIP --port=80
```

```
kubectl -n production expose deployment nginx-app --name=front-end-service-np
--type=NodePort --port=80
```

8. PODS and Namespaces

i) Create the namespace

```
kubectl create namespace website-frontend
```

ii) Launch the POD in the namespace

```
kubectl run jenkins --image=jenkins --restart=Never --namespace=website-frontend
```

9. Deployments

i) Create the directory to save the output

```
mkdir -p /opt/KUAL00201/
```

ii) Create the deployment

```
kubectl run kua100201 --image=redis --replicas=7 --restart=Always
--labels=app_env_stage=dev --dry-run -o yaml >/opt/KUAL00201/deploy_spec.yaml
```

iii) Delete the API Objects

```
kubectl delete deployment kua100201
```

10. Labels and Selectors

```
mkdir /opt/KUCC00302/
touch /opt/KUCC00302/kucc00302.txt
kubectl get pod -n production -l front-end-service > /opt/KUCC00302/kucc00302.txt
```

11. Secrets

i) Create the secret

kubectl create secret generic super-secret --from-literal=credential=alice
--from-literal=username=bob

ii) Create a POD with mounted secret via the file

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-secrets-via-file
spec:
  containers:
  - name: mypod
    image: nginx
    volumeMounts:
    - name: foo
      mountPath: /secrets"
  volumes:
  - name: foo
    secret:
      secretName: super-secret
```

iii) Create a POD with mounted secret via ENV

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-secrets-via-env
spec:
  containers:
  - name: mycontainer
    image: nginx
    env:
      - name: SUPERSECRET
        valueFrom:
          secretKeyRef:
```

```
                    name: super-secret
                    key: username
              - name: USER
                valueFrom:
                  secretKeyRef:
                    name: super-secret
                    key: password
          restartPolicy: Never
```

Documentation Link For Reference of both the above example templates:

https://kubernetes.io/docs/concepts/configuration/secret/

12. Volumes

i) Create the namespace

```
kubectl create namespace pre-prod
```

ii) Create the POD with volume

```
        apiVersion: v1
        kind: Pod
        metadata:
          name: non-persistent-redis
          namespace: pre-prod
        spec:
          containers:
          - image: redis
            name: redis
            volumeMounts:
            - mountPath: /data/redis
              name: cache-control
          volumes:
          - name: cache-control
            emptyDir: {}
```

13. Scaling Deployments

```
kubectl scale -n production deployment nginx-app --replicas=6
```

14. Metric Server

```
kubectl top pod --sort-by=cpu
```

15. DNS

```
kubectl run nginx-dns --image=nginx --restart=Always
kubectl expose deployment nginx-dns --port=80 --name=nginx-dns
kubectl run dnscheck --image=busybox:1.28 --command sleep 3600 --restart=Never
kubectl exec -it dnscheck nslookup nginx-dns > /opt/service.dns
kubectl get pods -o wide
kubectl exec -it dnscheck nslookup 10-40-0-5.default.pod > /opt/pod.dns
```

16. Node Draining

```
kubectl drain node01 --ignore-daemonsets --force
```

17. Persistent Volumes - Host Path

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/opt/pvsort.txt"
```

Reference Documentation:

https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

18. Sorting Operation

Download and create objects which are part of yaml

i) Perform the sorting operation

```
kubectl get pv --sort-by=.spec.capacity.storage >/opt/my_volumes.txt
```

19. Labels and Selectors

```
kubectl run kplabs-jenkins --image=jenkins --restart=Never
--labels=env=development,org=kplabs
```

20. Sorting Operation

```
kubectl get pv --sort-by=.metadata.name >/opt/pv-sort-name.txt
```

21. Deployment

```
kubectl run kplabs-nginx-deploy --image=nginx --replicas=2 --labels=org=kplabs
--restart=Always
```

22.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: cron-parallel
spec:
  parallelism: 2
  completions: 20
  template:
    metadata:
      name: cron-parallel
    spec:
      containers:
      - name: c
        image: busybox
        command: ["sh", "-c", "echo Hello CKA"]
      restartPolicy: OnFailure
```

23. POD Security Context

kubectl create namespace security

ii) Pod Security Context

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-security
  namespace: security
spec:
  securityContext:
    runAsUser: 1005
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "ping 127.0.0.1" ]
```

iii) Output the logs of the Pod

```
kubectl logs pod-security -n security > /tmp/pod-security.txt
```

24. Configure Cluster

Reference the original kubeadm videos for the same.

25. Taints

Reference this forum post in the exam

https://discuss.kubernetes.io/t/how-to-extract-the-list-of-nodes-which-are-tainted-unable-to-find-node-name-when-using-jsonpath-as-effect-noschedule-or-viceversa-in-the-kubernetes-command-line/8335

Use the last solution and make sure to replace ` with '

26. Static Pods

i) Modify the kubelet to add the static path

```
nano /etc/kubernetes/kubelet
```

Add the following contents:

```
--pod-manifest-path=/etc/kubelet.d/
```

ii) Go to the /etc/kubernetes/manifests/ directory and create manifest file

```
cd /etc/kubernetes/manifests/
```

```
kubectl run kplabs-static --image=nginx --restart=Never --dry-run -o yaml > kplabs-static.yaml
```

iii) Restart Kubelet

```
systemctl restart kubelet
```

27. Taints

```
kubectl taint node node01 mykey=mvalue:NoSchedule
```

28. Tolerations

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    run: kplabs-tolerate
  name: kplabs-tolerate
spec:
  replicas: 6
  selector:
    matchLabels:
      run: kplabs-tolerate
  template:
```

```
.                 metadata:
.                   creationTimestamp: null
.                   labels:
.                     run: kplabs-tolerate
.                 spec:
.                   containers:
.                   - image: nginx
.                     name: kplabs-tolerate
.                   tolerations:
.                   - key: "mykey"
.                     operator: "Equal"
.                     value: "myvalue"
.                     effect: "NoSchedule"
```